CONTINUATION APPLICATION

UNDER 37 CFR § 1.53(B)

TITLE: **AN UNSOLICITED MESSAGE REJECTING COMMUNICATIONS PROCESSOR**

APPLICANT(S): Richard Paul White, et al.

Correspondence Enclosed:

Utility Application Transmittal Sheet (1 pg.); FY 2004 Fee Transmittal Sheet (1 pg.); Specification, Claims, and Abstract (34 pgs); Formal Drawings (21 sheets); Declaration (4 pgs.); Application Data Sheet (3 pgs.); Nonpublication Request under 35 U.S.C. 122(b)(2)(B)(i) (5 pgs.); Check for $385.00; and Return Postcard

PRIORITY DATA: Under U.S.C., 37 CFR § 1.53(b) this application claims the benefit of U.S. Utility Patent Application Serial 10/238,812, filed September 9, 2002.

"EXPRESS MAIL" Mailing Label Number EU 988129137 US Date of Deposit January 20, 2004
I hereby certify under 37 CFR §1.10 that this correspondence is being deposited with the United States Postal Service as "Express Mail Post Office to Addressee" with sufficient postage on the date indicated above and is addressed to the Commissioner for Patents, PO BOX 1450, Alexandria VA 22313-1450.

Haw-minn Lu (Applicant)

SPECIFICATION

AN UNSOLICITED MESSAGE REJECTING COMMUNICATIONS PROCESSOR

RELATED APPLICATIONS INFORMATION

[001] This application is a continuation of Application Serial No. 10/238,812 entitled "An Unsolicited Message Rejecting Communications Processor," filed on September 9, 2002, which is incorporated herein by reference in its entirety as if set forth in full.

[002] This invention is related to co-pending applications, Application Serial No. 10/238,076 entitled "An Unsolicited Message Diverting Communications," filed on September 9, 2002 by the same inventors and Application Serial No. 10/238,216 entitled "An Unsolicited Message, Intercepting Communications Processor," filed on September 9, 2002 by the same inventors.

BACKGROUND OF THE INVENTION

1. Field of the Inventions

[003] This invention blocks unsolicited e-mail messages, commonly known as SPAM, from a client's e-mail server; while reducing Internet traffic, local LAN traffic, and local storage.

2. Background Information

[004] E-mail has become an important means of communications. Unfortunately, unsolicited e-mail messages, commonly referred to as SPAM, is cluttering this communications channel. This unsolicited e-mail wastes Internet bandwidth, local area network bandwidth, and storage. This translates into lost productivity, increased

computing and increased communication costs. Some of this unsolicited e-mail is also offensive and inappropriate for non-adult audiences.

[005] The spammer collects a list of e-mail address, append these addresses to their message, and queues these messages on their e-mail server (message transfer agent), **2**, in **Fig. 1**. He then attaches their e-mail server to the Internet, **1**, via in a rogue Internet service provider, a dial-up connection, a digital scriber loop (DSL) connection, or a cable modem connection and sends out their message to the gateway message transfer agents, **5**, associated with each e-mail address. These gateway message transfer agent either stores the message in an e-mail mailbox associated with the client, **3**, or forwards the message to a another message transfer agent (MTA) on the same local area network.

[006] There are four basic approaches to trying to detect junk e-mail messages. One approach used a community set of rules to determine whether or not a message is spam. This approach is used in Razor, an open source Linux solution, and by companies such as CloudMark (based on Razor) and SpamNet. The problem is getting a community of users to agree on a common set of rules.

[007] A second approach uses a set of rule base filters which are periodically updated by the provider and downloaded by the client to determine whether or not a message is spam. The problem is that the set of rules have to be updated and downloaded periodically.

[008] A third approach uses a set of permissions to determine whether or not a message is spam. The problem is that it is impossible for somebody not on a user's permission list to send a message to the user.

[009] A fourth approach uses a "whitelist" and a "blacklist" to determine whether or not a message is spam. The problem is that the spammers are constantly changing their return address and declared domain names.

[010] There are three basic ways of implementing these approaches. One implementation approach is in the Message transfer agent. This approach adds some rules to the MTA. The problem is that the MTA program is complicated and inflexible. This limits the kind of rules that can be implemented.

[011] A second implementation approach involves placing the filters between the e-mail client and the Message Transfer Agent. The problem is that some of the information which can be used to help determine whether or not a message is spam is lost or buried.

[012] A third implementation approach involves adding some filters to the e-mail client, Mail User Agent (MUA). The problem is that the e-mail client add-in interface is not an open standard. This causes compatibility problems.

[013] One problem with these approaches is that they are "reactive." The spam has already been received by the server and relayed via a local area network to client's computer. The spam message has already consumed the server's Internet bandwidth, local area network bandwidth, and client storage.

[014] Another problem with many of these approaches is that they are based on the from-address, the subject line, or content of the message; all of which can be easily forged or changed.

OBJECTS AND ADVANTAGES

[015] Accordingly, the several objects and advantages of my invention are:

a) to provide a procedure which eliminates unsolicited messages from a client's e-mail mailbox;

b) to provide a procedure which reduces the amount of the client's Internet bandwidth consumed by unsolicited messages;

c) to provide a procedure which reduces the amount of communications bandwidth between a client's message transfer agent and mail user agent consumed by unsolicited messages;

d) to provide a procedure which reduces the amount of storage consumed by unsolicited messages;

e) to provide a procedure which uses information which can not be forged to improve the ability to block unsolicited messages;

f) to provide a procedure which avoids the need for users to install software on their individual system;

g) to provide a procedure which eliminates any need to change the client's current message transfer agent;

h) to provide a procedure which logs the messages which have been allowed;

i) to provide a procedure which logs the messages which have been rejected;

j) to provide a procedure which allows users to select the ability to not block unsolicited messages;

k) to provide a procedure which allows users to select the ability to block unsolicited messages;

l) to provide a procedure which gives feedback to the sender of a blocked message;

m) to provide a procedure for blocking unsolicited messages which is failsafe; and

n) to provide a procedure for blocking unsolicited messages which is scalable.

[016] Further objects and advantages of our invention will become apparent from a consideration of the drawings and ensuing description.

SUMMARY OF THE INVENTION

[017] The spam blocker monitors the SMTP/TCP/IP conversation between a sending message transfer agent MTA_0 and a receiving message transfer agent MTA_1 and catches the IP address IP_0 of MTA_0; the declared domain D_0 of MTA_0; the from-address A_0; and the to-address A_1.

[018] The spam blocker uses the captured information to run various source and content based tests. IP_0 is used to query a DNS server for the real domain name DD_0. DD_0 is tested to see if it is "no name." IP_0 is tested to see if it is in a open relay database. IP_0 is tested to see if it is not in a allow_address database. IP_0 is tested to see if it is in a prevent_address database. DD_0 and MTA_1's domain name D_1 are tested to see if they match. D_0 is tested to see if it does not match DD_0 and if D_0 is in the suspect_domain database. A_0 is tested to see if any portion of it is in the bad_from database. The domain of A_0 is tested to see if does not match DD_0 and if it is in the suspect_domain database. A_1 is tested to see if it is in the no_filter database. A_1 is tested to see if it is not in the yes_filter database. A_1 is tested to see if it matches A_0.

[019] The spam blocker interrupts the conversation between MTA_0 and MTA_1 when MTA_0 sends a RCPT command and uses the results of the various tests to decide if the message is suspected of being unsolicited.

[020] If the message is suspected of being unsolicited then the spam blocker logs the rejected message in a rejected_connection database and sends an permanent negative completion reply to MTA_0 which forces MTA_0 to terminate the connection with MTA_1 before the body of the message is transmitted.

[021] If the message is not suspected of being unsolicited then the spam blocker logs the allowed message in the allowed_connection database, releases the intercepted RCPT command, and allows the conversation between MTA_0 and MTA_1 to proceed.

BRIEF DESCRIPTION OF THE DRAWINGS

[022] **Fig. 1** (prior art) shows a overview block diagram of a networked computer system consisting of a network **1**, a client, a transmitting message transfer agent **2**, a receiving message transfer agent **5**, and a open relay message transfer agent **6**.

[023] **Fig. 2** shows a block diagram of a typical e-mail transfer between sending user **8**, sending mail user agent **9**, sending message transfer agent **11**, a spam blocker **18**, a receiving message transfer agent **12**, a receiving local message transfer agent **14**, receiving mail user agent **16**, and receiving user **16**.

[024] **Fig. 3** (prior art) shows a list of simple mail transfer protocol (SMTP) commands and their associated reply codes.

[025] **Fig. 4** (prior art) shows a list of simple mail transfer protocol (SMPT) reply codes and what they mean.

[026] **Fig. 5** (prior art) shows a timeline of a typical message transfer using the simple mail transfer protocol (SMTP).

[027] **Fig. 6** shows a timeline of a modified version of the simple mail transfer protocol used to reject unsolicited messages in which the message is determined not to be spam.

[028] **Fig. 7** shows a timeline of a modified version of the simple mail transfer protocol used to reject unsolicited messages in which the message is determined to be spam.

[029] **Fig. 8A–8F** shows a flowchart of the spam blocking algorithm used to reject unsolicited messages.

[030] **Fig. 9** shows a flowchart for a domain name server (DNS) database server query to fetch the domain name IP_0 associated with a IP address of the sending message transfer agent.

[031] **Fig. 10** shows a flowchart for the IP_resolve test which checks if DNS database has a domain name DD_0 associated with a IP address IP_0 of the sending message transfer agent.

[032] **Fig. 11** shows a flowchart for the open_relay test which checks if the IP address IP_0 of the sending message transfer agent is in an open relay database.

[033] **Fig. 12A** shows a flowchart for the allow_address test which checks if IP address IP_0 of transmitting message transfer agent is in the allow_address database.

[034] **Fig. 12B** shows the data structure of the allow_address database which stores the IP address of trusted message transfer agents.

[035] **Fig. 12C** shows a sample of the contents of the allow_address database.

[036] **Fig. 13A** shows a flowchart for the prevent_address test which checks if the IP address IP_0 of transmitting message transfer agent is in the prevent_address database.

[037] **Fig. 13B** shows the data structure of the prevent_address database which stores the IP addresses of known spammers and of dial-in, digital scriber loop, and cable modem accounts from which a message transfer agent should not be operating.

[038] **Fig. 13C** shows a sample of the contents of the prevent_address database.

[039] **Fig. 14** shows a flowchart for the echo_domain test which checks if the declared domain D_0 matches the domain of the receiving message transfer agent D_1.

[040] **Fig. 15** shows a flowchart for the forged_domain test which checks if the declared domain D_0 does not match the real domain DD_0 of the sending message transfer agent and if the declared domain D_0 of the sending message transfer agent is in the suspect_domain database.

[041] **Fig. 16A** shows a flowchart for the bad_from test which checks if any portion of the from-address is in the bad_from database.

[042] **Fig. 16B** shows the data structure of the bad_from database which stores some portions of commonly used from-addresses used by spammers.

[043] **Fig. 16C** shows a sample of the contents of the bad_from database.

[044] **Fig. 17A** shows a flowchart for the suspect_domain test which checks if the real domain DD_0 of the sending message transfer agent does not match the domain name of the from-address A_0 and domain of the from-address A_0 is in the suspect_domain database.

[045] **Fig. 17B** shows the data structure of the suspect_domain database which stores domain names commonly forged by spammers.

[046] **Fig. 18** shows a flowchart for the to_from test which checks if the to-address A_1 of the message matches the from-address A_0.

[047] **Fig. 19A** shows a flowchart for the no_filter test which checks if the to-address A_1 of the message in the no_filter database.

[048] **Fig. 19B** shows the data structure of the no_filter database which stores the to-addresses A_1 that do not want to be filtered.

[049] **Fig. 20A** shows a flowchart for the yes_filter test which checks if the to-address A_1 of the message in the yes_filter database.

[050] **Fig. 20B** shows the data structure of the yes_filter database which stores the to-addresses A_1 that do want to be filtered.

[051] **Fig. 21** shows the data structure of the completed_connection database which stores the time and to-address of the message.

[052] **Fig. 22** shows the data structure of the rejected_connection database which stores the time, to-address, from-address, and a code which indicates the reason that the connection was rejected.

DESCRIPTION OF INVENTION

[053] In overview. a user, **8**, in **Fig. 2**, sits at a terminal and types an e-mail message into a mail user agent (MUA), **9**, such as Microsoft Outlook Express or Eudora. This mail message is then sent to a mail queue, **10**, associated with a message transfer agent (MTA), **11**, such as Unix Sendmail. MTA, **11**, contacts MTA, **12**, associated with the to-address of the mail message and negotiates the transfer of the queued mail message. If the negotiations are completed in a satisfactory manner, the message is transferred and stored in a queue, **13**, associated with the gateway MTA, **12**. MTA **12** then either stores the mail message in the user's mailbox, **15**, or relays the mail message to another MTA, **14**, associated with the user. The receiving user, **16**, then uses their mail user agent (MUA), 17, to fetch the mail message from their mailbox **15**.

[054] The present invention, a spam blocker, **18**, in **Fig. 2**, is inserted between MTA **11** and MTA **12**. The spam blocker monitors the conversation between MTA **11** and MTA **12**, uses some of the transferred information to decide if the message is spam, and manipulates the conversation if the message is determined to be spam. The spam blocker operates at the SMTP protocol level. This increases the accuracy of spam determination since more information is available. This reduces compatibility issues since SMTP is an established standard and since there are no other in-band or out-of-band communication channels between MTA **11** and MTA **12** to deal with. Operating at the protocol level increases the type of responses available for dealing with the unsolicited mail messages.

[055] As mentioned previously, the operations of spam blocker is tightly coupled with the SMTP protocol. The SMTP protocol consists of a set of commands and a set of replies. The set of commands, HELO, MAIL, RCPT, DATA, RSET, SEND, SOML, SAML, VRFY, EXPN, HELP, NOOP, QUIT, and TURN is shown in **Fig. 3** (prior art) and the set of replies shown in **Fig. 4** (prior art) are detailed in RFC 821 (Postel 1982) and discussed in *TCP/IP Illustrated Volume 1, The Protocols* by W. Richards Stevens (Addison Wesley, ISBN 0-201—63346-9). The possible replies associated with each command is shown in **Fig. 3**. A "S" class reply indicates a successful completion of the command. A "F" class reply indicates a failure in completion of the command. A "E" class reply indicates an error in the completion of the command. A reply code beginning with a 1, 2, 3, 4, and 5 respectively indicate a positive preliminary reply, a positive completion reply, positive intermediate reply, transient negative completion reply, and permanent negative completion reply. The format of an mail message is specified in RFC 822 (Crocker 1982).

[056] As an example, the time line of a normal mail transfer is shown in **Fig. 5** (prior art). For purposes of explanation, assume that the sending MTA **11** is called MTA_0 and has an IP address of IP_0, a declared domain name of D_0, a real domain name DD_0, and the mail message has a from-address of A_0 and that the receiving MTA 12 is called MTA_1 and has an IP address of IP_1, a domain name of D_1, and it is associated with an user mail address of A_1.

[057] MTA_0 sends a message to MTA_1 requesting to establish a connection. MTA_1 then responds with a 220 rely which includes it's domain name D_1. MTA_0 then sends a HELO command with it's declared domain name D_0. MTA_1 then

responds with a 250 reply to acknowledge the successful completion of the HELO command. MTA_0 then sends a MAIL command with a message from-address A_0. MTA_1 then sends a 250 to acknowledge the successful completion of the MAIL command. MTA_0 then sends a RCPT command with a to-address A_1. MTA_1 then sends a 250 reply to acknowledge successful completion of the RCPT command. MTA_0 then sends a DATA command. MTA_1 then sends a 354 reply to acknowledge its ability to receive the data. MTA_0 then sends the data, the body of the message. MTA_0 then sends a .\r\n to indicate the end of the data. MTA_1 then sends a 250 to acknowledge successful transfer of the data. MTA_0 then sends a QUIT command. MTA_1 then sends a 221 to acknowledge the QUIT command and closes its end of the TCP connection. Upon receipt of the 221 reply MTA_0 closes its end of the TCP connection.

[058] As mentioned previously, the present invention, a spam blocker, **18,** in **Fig. 17,** is inserted between MTA_0 and MTA_1. The spam blocker monitors the conversation between MTA **11** and MTA **12**, uses some of the transferred information to decide if the message is spam, and manipulates the conversation if the message is determined to be spam. For purposes of illustration, assume that the spam blocker has an IP address of IP_2 and that the MX resource record associated with D_1 has been changed to point to IP address IP_2 instead of IP_1.

[059] Time lines illustrating the interactions between MTA_0, spam blocker, and MTA_1 are shown in **Fig. 6** and **Fig. 7**. MTA_0 begins by sending a message to D_1 requesting to establish a connection. The MX resource record directs this request to IP_2. The spam blocker at IP_2 notes the IP address IP_0 of MTA_0, relays this request to MTA_1, and performs various source based spam determining tests based on IP_0.

MTA_1 replies with a 220 acknowledgement reply which includes its domain name D_1. The spam blocker relays this acknowledgement to MTA_0. MTA_0 then sends a HELO command which includes declared domain name D_0. The spam blocker notes declared domain name D_0 , relays this command to MTA_1, and performs various source based spam determining tests based on D_0. MTA_1 replies with a 250 to acknowledge successful completion of the HELO command. The spam blocker relays this reply to MTA_0. MTA_0 then sends a MAIL command which includes from-address A_0 The spam blocker notes from-address A_0, relays this command to MTA_1, and performs various source and content based spam determining tests based on A_0. MTA_1 replies with a 250 to acknowledge successful completion of the MAIL command. The spam blocker relays this reply to MTA_0. MTA_0 then sends a RCPT command which includes to-address A_1. The spam blocker notes A_1 and performs various source and content based spam determining tests based on A_1. The spam blocker then evaluates the results of the various source and content based spam determining tests to determine whether or not the message is suspected of being spam.

[060] If the message is not suspected of being spam then the interactions between MTA_0, spam blocker, and MTA_1 proceed as shown in **Fig. 6**. The spam blocker relays the RCPT command to MTA_1. MTA_1 replies with a 250 to acknowledge successful completion of the RCPT command. The spam blocker relays the 250 reply to MTA_0. MTA_0 then sends a DATA command. The spam blocker relays the DATA command to MTA_1. MTA_1 replies with a 354 to indicate that the mail transfer may begin. The spam blocker relays the 354 to MTA_0. MTA_0 then sends the body of the mail message. The spam blocker relays the body of the mail message to

MTA_1. MTA_0 then sends a .\r\n to indicate the end of the message. The spam blocker relays the .\r\n to MTA_1. MTA_1 replies with a 250 to acknowledge receiving the end of message indicator. The spam blocker relays the 250 reply to MTA_0. MTA_0 then sends a QUIT command. The spam blocker relays the QUIT command to MTA_1. MTA_1 replies with a 221 reply to acknowledge the QUIT command and closes down its end of the TCP connection. The spam blocker relays the 221 reply to MTA_0 and closes its end of TCP connection to MTA_1 and MTA_0. Upon receiving the 221 reply, MTA_0 closes down its end of the TCP connection.

[061] If the message is suspected of being spam then the interactions between MTA_0, spam blocker, and MTA_1 proceed as shown in **Fig. 7**. The spam blocker sends a 550 error reply to MTA_0. MTA_0 replies to the 550 error reply with a QUIT command. The spam blocker relays the QUIT command to MTA_1. MTA_1 replies with a 221 and closes its end of the TCP connection. The spam blocker relays 221 reply to MTA_0 and closes down its end of the TCP connections to MTA_1 and MTA_0. Upon receiving the 221 reply MTA_0 closes down its end of the TCP connection.

[062] It should be noted that the spam blocker terminates the connection with MTA_0 before the body of the message is transferred. This saves Internet bandwidth, server-client bandwidth, and local storage. It should be noted that in the case where the message is determined not to be spam, the interactions between MTA_0 and MTA_1 as shown in **Fig. 6** are functionally equivalent to the normal interactions between MTA_0 and MTA_1 as shown in **Fig. 3** (prior art). This improves compatibility. It should be noted that until receipt of the RCPT command by the spam blocker, MTA_1 has full control of the replies to MTA_0's commands. This also improves compatibility.

[063] The algorithm which implements the spam blocker is shown in **Figures 8A, 8B, 8C, 8D, 8E,** and **8F**. Connections **50, 51,** and **52** in **Fig. 8A** are respectively connected to connections **55, 56,** and **57** in **Fig. 8B**. Connections **60**. **61**, and **62** in **Fig. 8B** are respectively connected to connections **65, 66,** and **67** in **Fig. 8C**. Connections **70, 71,** and **72** in **Fig. 8C** are respectively connected to connections **75, 76,** and **77** in **Fig. 8D**. Connections **80, 81,** and **82** in **Fig. 8D** are respectively connected to connections **85, 86, 87** in **Fig. 8E**. Connections **90, 91,** and **92** in **Fig. 8E** are respectively connected to connections **95, 96,** and **97** in **Fig. 8F**.

[064] The algorithm begins in **Fig. 8A** by waiting for a connection establishment request. Upon receipt of such a request it extracts MTA_0's IP address IP_0. It then sends a 220 reply with a D_1 domain name to MTA_0. It then requests a domain name server (DNS) pointer query. It performs a IP_0_resolve test, an open relay test, an allow_address, and a prevent_address test.. It then request a connection with MTA_1 and waits for a 220 reply from MTA_1. It then waits for a reply from either MTA_0 or MTA_1. It should be noted that IP_0 is an important address since it cannot be forged. If IP_0 is forged then the 220 reply will not be sent to MTA_0's real IP address, MTA_0 will not receive the 220 reply, and the requested connection will not be established.

[065] If the reply is from MTA_1 it relays the reply to MTA_0 and waits for another reply.

[066] If the reply is a HELO command as shown in **Fig. 8B** then the algorithm extracts MTA_0's declared domain D_0. It then performs an echo_domain and forged_domain tests. It then relays the HELO reply to MTA_1 and waits for another reply.

[067] If the reply is a MAIL command as shown in **Fig. 8C** then the algorithm extracts from-address A_0. The algorithm then performs the bad_from and the suspect_domain tests. It then relays the MAIL command to MTA_1 and waits for another reply.

[068] If the reply is a RCPT command as shown in **Fig. 8D** then the algorithm extracts to-address A_1. It then performs the no_filter, the yes_filter, and the to_from tests. It then uses the results of the various test to determine if the message is suspected of being spam. In this version of the algorithm the decision equation used is t_allow OR t_no_filter OR NOT t_yes_filter OR NOT (t_prevent OR t_open_relay OR t_IP_resolve OR t_bad_from OR t_suspect_domain OR t_to_from OR t_echo_domain OR t_forged_domain).

[069] If the decision equation is determined to be true then the message is allowed and the time and to-address is entered in the allowed_connection database. The structure of the allow_connection database is shown in **Fig. 21**. It is anticipated that this database will be used for statistical and billing purposes. The from-address A_0 could be included if privacy issues are not of concern. It then relays the RCPT command to MTA_1 and waits for a new reply.

[070] If the decision equation is determined to be false then the message is not allowed and the time, from-address A_0, to-address A_1, and a reason for the rejection is entered into a rejected_connection database. The data structure of the rejected_connection database is shown in **Fig. 22**. It is anticipated that this database will be used for statistical and billing purposes. The algorithm then rejects the MTA_0 connection by sending a 550 error reply to MTA_0 and waits for a new reply.

[071] If the reply is a DATA command as shown in **Fig. 8E** then the algorithm relays the DATA command to MTA_1, waits for a 354 reply form MTA_1, and relays the 354 reply to MTA_0. It then waits for data from MTA_0, relays this data to MTA_1, waits for a .\r\n end of message indicator from MTA_0, relays the .\r\n to MTA_1, waits for a 250 reply from MTA_1 to acknowledge the successful transfer of the data, and then relays the 250 reply to MTA_0 and waits for a new reply.

[072] If the reply is a RSET, SEND, SCML, SAML, VRFY, NOOP, EXPN, HELP, or TURN command as shown in **Fig. 8F** then the algorithm relays the command to MTA_1 and waits for a new reply.

[073] If the reply is a QUIT command as shown in **Fig. 8F** then the algorithm relays the QUIT command to MTA_1, waits for a 221 reply from MTA_1 to acknowledge the QUIT command, relays the 221 reply to MTA_0, closes down its end of the TCP connection with both MTA_1 and MTA_0, and waits for a new connection request.

[074] If the reply is not a QUIT command then this is invalid command error and the algorithm sends a 500 to MTA_0 and waits for a new reply.

[075] The algorithm for requesting the domain name server (DNS) pointer query in **Fig. 8B** is shown in **Fig. 9**. The algorithm begins by starting a timer. It then sends out a pointer query to a DNS server to fetch the domain name associated with IP_0. It then waits for either a reply or timer time out. If the timer times out then the real domain DD_0 is set to the declared domain D_0. If the DNS server replies with a domain name and the timer did not time out then the real domain DD_0 is set to the returned domain

name. If the DNS reply indicates that it does not have a domain name associated with IP_0 and the timer did not time out then the real domain DD_0 is set to "no name."

[076] The algorithm for the IP resolve test in **Fig. 8B** is shown in **Fig. 10**. If real domain name DD_0 is "no name" then set flag t_IP_resolve to true, else set flag t_IP_resolve to false. This test is based on the fact that some spammers try to hide their domain name and do not enter a domain name in the DNS database.

[077] The algorithm for the open relay test in **Fig. 8B** is shown in **Fig. 11**. The algorithm begins by starting a timer and then checking to see if IP_0 is in an open relay database, such as http://www.relays.ordb.org or http://www.relays.orisrusoft.com. It then waits for either a reply from the open relay database or a timer time out. If the timer does time out then set flag t_open_relay to false. If IP_0 is in an open relay database and the timer does not time out then set flag t_open_relay to true, else set flag t_open_relay to false. This test is based on the fact that many spammers try to hide their real IP address by sending their spam to an open relay MTA. This is shown in **Fig. 1** The spammer's MTA, **2**, first sends the unsolicited message to open relay MTA, **6**. MTA **6** then contacts client's MTA, **5**, to relay the mail. Client MTA **5** then thinks that the received mail is from open relay MTA, **6**, rather than spamming MTA, **2**.

[078] The algorithm for the allow_address test in **Fig. 8B** is shown in **Fig.12A**. If IP_0 is in the allow_address database then set flag t_allow to true, else set flag t_allow to false. The data structure of the allow_address database is shown in **Fig. 12B**. This database contains the IP address of trusted message transfer agents, such as those associated with AOL, Earthlink, ATT, etc. A sample of some of the allow_address database entries is shown in **Fig. 12C**.

[079] The algorithm for the prevent_address test in **Fig. 8B** is shown in **Fig. 13A**. If IP_0 is in the prevent_address database then set flag t_prevent to true, else set flag t_prevent to false. The data structure of the prevent_address database is shown in **Fig. 13B**. This database contains the IP address of known spammers. This database also contains MTA's that are not RFC compliant, such as those listed in http://www.rfc.ignorant.org . This database also contains blocks of suspicious IP addresses such as those associated with dial-in, digit scriber loop (DSL), and modem connections since there is no valid reason for a MTA to originate from one of these IP addresses. A sample of some of the prevent_address database entries is shown in **Fig. 13C**.

[080] The algorithm for the echo_domain test in **Fig. 8B** is shown in **Fig. 14**. If declared domain D_0 does match the domain D_1 of MTA_1 then set flag echo_domain to true, else set flag echo_domain to false. This test is based on the fact that some spammers set their declared domain to a domain name trusted by MTA_1.

[081] The algorithm for the forged_domain test in **Fig. 8B** is shown in **Fig. 15**. If the declared domain D_0 does not match the real domain name DD_0 and the declared domain D_0 is in the suspect_domain database then set flag t_forged_domain to true, else set flag t_forged_domain to false. The data structure of the suspect_domain database is shown in **Fig. 17B**. This test is based on the fact that some spammers hide their true domain name and assume popular domain names such as yahoo.com or hotmail.com however their declared domains D_0 does not match their real domain name DD_0.

[082] The algorithm for the bad_from address test in **Fig. 8C** is shown in **Fig. 16A**. If A_0 is in the bad_from database then set flag t_bad_from to true, else set flag

t_bad_from to false. This test is based on the fact that some spammers use variations of the same from-address for different spam messages. The data structure of the bad_from database is shown in **Fig. 16B**. Some examples of bad_from database entries are shown in **Fig. 16C**. The * symbol indicates at least one wildcard character.

[083] The algorithm for the suspect_domain test in **Fig. 8C** is shown in **Fig. 17A**. If the real domain DD_0 does not match the domain of the from-address A_0 and the domain of the from_address A_0 is in the suspect_domain database then set flag t_suspect_domain to true, else set flag t_suspect_domain to false. The data structure of the suspect_domain database is shown in **Fig. 17B**. This test is based on the fact that some spammers hide their true domain name and assume popular domain names such as yahoo.com or hotmail.com however the domain of their from-address A_0 does not match their real domain name DD_0.

[084] The algorithm for the to_from test in **Fig. 8D** is shown in **Fig. 18**. If from-address A_0 matches to-address A_1 then set flag t_to_from to true, else set flag t_to_from to false. This test is based on the fact that some spammers try to disguise themselves as someone known to MTA_1, such as A_1.

[085] The algorithm for the no_filter test in **Fig. 8D** is shown in **Fig. 19A**. If to-address A_1 is in the no_filter database then set flag t_no_filter to true, else set flag t_no_filter to false. The data structure of the no_filter database is shown in **Fig. 19B**. This test is based on the fact that some users do not want their mail filtered.

[086] The algorithm for the yes_filter test in **Fig 8D** is shown in **Fig. 20A**. If to-address A_1 is in the yes_filter database then set flag t_yes_filter to true, else set flag

t_yes_filter to false. The data structure of the yes_filter database is shown in **Fig. 20B**. This test is based on the fact that some users want their mail filtered.

[087] The spam blocking process can be scaled to handle more messages by deploying more spam blocking units each connected to a client message transfer agent and changing the client's MX record to distribute the mail traffic among all the spam blocking units.

[088] The spam blocking process can be made failsafe by deploying more spam blocking units each connected to a client message transfer agent and changing the client's MX record to distribute the mail traffic among all the spam blocking units and not to distribute mail traffic to any spam blocking unit which is off-line.

[089] Objectives of this invention are to reduce the Internet bandwidth consumed by unsolicited messages, to reduce the communications bandwidth consumed between the client's message transfer agent and mail user agent, to reduce the amount of storage consumed by unsolicited messages, and to eliminate unsolicited messages from a client's e-mail box. This is accomplished by rejecting an unsolicited message before the body of the message is transferred to the client's message transfer agent.

[090] Another objective of this invention is to improve the ability to detect unsolicited messages. This is accomplished by using the sending MTA's IP address, IP_0 which can not be forged. The problem is that content based tests suffer from the fact that the declared domain, from-address, subject line, or message body can easily be forged or changed. IP_0 can not be forged because a forged IP_0 would cause the receiving MTA's 220 rely to be sent to the wrong address and without a 220 reply the sending MTA would never be able to establish the requested connection. Source based tests are thus based on

"bedrock" information while content based tests are based on "shifting sands." This improves the ability of the spam blocker to detect suspected unsolicited messages.

[091] Other objectives of this invention are to avoid the need for users to install software on their individual systems and the need to change the client's current message transfer agent. This is accomplished by positioning the spam blocker between the sending and receiving message transfer agents and not between the receiving message transfer agent and the mail user agent. This is further accomplished by not positioning the spam blocker as a message transfer agent add-on or mail user agent add-on. This is further accomplished by keeping the communications between the spam blocker and the sending and receiving message transfer agents standard SMTP protocol and eliminating other in-band or out-of-band communications channels between the spam blocker and the sending message transfer agent, receiving message transfer agent, or the mail user agent.

[092] Another objective of this invention is to log the messages that have been allowed or rejected. This is accomplished by the allow_connection and the rejected_connection databases. This is further accomplished by deferring the decision equation until after the to-address A_1 has been transmitted and captured.

[093] Another objective of this invention is to allow users to select the ability not to block unsolicited messages. This is accomplished by the no_filter test and the no_filter database. A related objective of this invention is to allow users to select the ability to block unsolicited messages. This is accomplished by the yes_filter test and the yes_filter database.

[094] Another objective of this invention is to give feedback to the sender of a blocked message. This is accomplished by the sending a permanent negative completion

reply such as a 550 reply to the sending message transfer agent if a message is determined to be unsolicited. The sending message transfer agent will then either send a error message to the sender or send the message back to the message transfer agent that sent the message. This is important since most spam blocking agents either delete or divert the spam into a junk folder and does not return an error message. The sender of a false positive unsolicited message would thus have no knowledge that the message was not delivered.

[095] Other objectives of this invention are to scale the spam blocking process and to make the spam blocking process failsafe. This is accomplished by deploying more spam blocking units each connected to a client message transfer agent and changing the client's MX record to distribute the mail traffic among all the spam blocking units and not to distribute mail traffic to any spam blocking unit which is off-line.

[096] The spam blocker algorithm shown in **Figures. 8 A – 8F** is structured for clarity rather than efficiency. It is assumed that anyone skilled in the art can restructure spam blocker algorithm for efficiency The DNS and open relay database server time out problems can be avoided by setting up a local copies of the servers. All, none or any subset of the tests shown in **Fig. 8A** can be used in any order at any time before the decision equation shown in **Fig. 8D**. All, none, or any subset of the tests shown in **Fig. 8B** can be used in any order at any time before the decision equation shown in **Fig. 8D**. All, none, or any subset of the tests shown in **Fig. 8C** can be used in any order at any time before the decision equation shown in **Fig. 8D**. More tests can be added. The decision function has to be modified if any of the tests are not used or more tests are added. The decision equation can be deferred until MTA_1's 250 reply to the RCPT

command. This still prevents the body of the message from being transmitted since the decision equation still proceeds MTA_0's DATA command. While a 550 reply was used to force MTA_0 to QUIT the connection, other 5XX permanent negative completion replies could also be used.

[097] Although the present invention has been described above in terms of specific embodiments, it is anticipated that alteration and modifications thereof will no doubt become apparent to those skilled in the art. It is therefore intended that the following claims be interpreted as covering all such alterations and modifications as falling within the true spirit and scope of the invention.